UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/867,650	05/31/2001	Michael Anthony Sijacic	06502.0338-00000	7983

60667 7590 01/30/2007

SUN MICROSYSTEMS/FINNEGAN, HENDERSON LLP
901 NEW YORK AVENUE, NW
WASHINGTON, DC 20001-4413

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	**PAPER NUMBER**
2195	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	01/30/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	09/867,650	SIJACIC ET AL.	
	Examiner	Art Unit	
	Lewis A. Bullock, Jr.	2195	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 October 2006*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-91* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-91* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *8/9/06*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

2. Claims 1-91 are rejected under 35 U.S.C. 102(e) as being anticipated by CHONG (U.S. Patent Application Publication 2002/0184610).

As to claim 1, CHONG teaches a method for creating an activity (component / object / workflow step) within a process management system (development system / workflow environment), comprising: receiving first data reflecting a class file (class); receiving second data reflecting a data representation file (imported media / variables); packaging the first and second data (via receiving the classes and variable information packaged in a Jar file) (via componentization) (pg. 29, paragraph 431-434); and associating the packaged data with an activity (via all information are stored in a Jar file) that may be used in an automated workflow process (workflow) (via dragging and dropping the icon representing the component to a diagram of the workflow) (pg. 13, paragraph 257 – 258; pg. 14, paragraph 262) to access information external to the process management system (via using the adapters to access the external data

sources and manipulate the external data sources) (pg. 23, paragraph 350 – pg. 24, paragraph 352; pg. 24, paragraph 353-354).

As to claim 12, CHONG teaches a method for implementing a custom activity (component / object /workflow step) within a process management environment (development system / workflow environment), comprising: defining a file associated with a custom activity (via the user creating a component and the class of the component via componentization) (pg. 29, paragraph 430-434); assigning a visual representation associated with the custom activity (via packaging the visual icon via componentization) (pg. 29, paragraph 430-434); receiving an indication (via drag and drop operation) reflecting implementation of the custom activity in a workflow process based on a position of the visual representation in a process map (workflow diagram) representing the workflow process (via dragging and dropping the icon representing the component to a diagram of the workflow and linking the instantiated component) (pg. 13, paragraph 257 – 258; pg. 14, paragraph 262); and invoking the file (via sending and loading the file and allowing the component to be used in an application) (pg. 30, paragraph 433).

As to claim 14, CHONG teaches a method for creating and defining a custom activity (component / object / workflow step) within a process management system (development system / workflow environment), comprising: creating at least one file defining properties (various attributes of the component, i.e. items i-viii) associated with

the custom activity (pg. 29, paragraph 430-434); and defining a model (visual icon) associated with the custom activity (via packaging the visual icon via componentization) (pg. 29, paragraph 430-434), wherein the custom activity may be used to access information external to the process manager system (via using the adapters of the component to access the external data sources and manipulate the external data sources) (pg. 23, paragraph 350 – pg. 24, paragraph 352; pg. 24, paragraph 353-354).

As to claim 20, CHONG teaches a method for implementing a custom activity (component / object / workflow step) in a process management system (development system / workflow environment), comprising: creating a process map (workflow diagram) reflecting an automated workflow process (pg. 13-14, paragraph 257-259); creating an image (visual icon) reflecting a custom activity (via packaging the visual icon with the component such that when the component is loaded the icon is displayed) (pg. 29, paragraph 430-434; pg. 14, paragraph 258); and invoking a class (class) defining the custom activity based on a manipulation of the image by a user (via the drag and drop operation) such that the image is placed in the process map (via drag and dropping the component into the workflow diagram) (pg. 13-14, paragraph 257-259), wherein the custom activity exchanges data with resources external to the process management system (via using the adapters of the component to access the external data sources and manipulate the external data sources) (pg. 23, paragraph 350 – pg. 24, paragraph 352; pg. 24, paragraph 353-354).

As to claim 21, CHONG teaches a method for creating a custom activity (component / object / workflow step) in a process management system (development system / object / workflow step), the custom activity exchanging information with resources external to the process management system (via using the adapters of the component to access the external data sources and manipulate the external data sources) (pg. 23, paragraph 350 – pg. 24, paragraph 352; pg. 24, paragraph 353-354), comprising: receiving a first file and a second file; and archiving the files in an archive file (via receiving the classes and variable information packaged in a Jar file) (via componentization) (pg. 29, paragraph 431-434) such that when the custom activity is activated the archived files are accessed and executed (via sending and loading the Jar file and allowing the component to be used when dragged into the workflow diagram in an application) (pg. 30, paragraph 433).

As to claim 27, CHONG teaches a memory for storing data for access by a process (application) being executed by a processor, the memory comprising: a structure defining a class file (class) and a data representation file (imported media / variables), packaging the files (via receiving the classes and variable information packaged in a Jar file) (via componentization) (pg. 29, paragraph 431-434), assigning an icon (visual icon) representing the packaged files (component) (via the visual icon is part of the Jar file) (pg. 29, paragraph 431-434), and associating the icon with an activity that performs processes defined by the class and data representation files (via drag and

dropping the component into the workflow diagram by manipulating the visual icon) (pg. 13-14, paragraph 257-259).

As to claim 30, CHONG teaches a memory for storing data for access by a process (application) being executed by a processor, the memory comprising: a structure for maintaining an identity of a custom activity (component / object / workflow step), parameters associated with the custom activity (parameters), a first hashtable reflecting data values to be used as input argument in a method, and a second hashtable reflecting output arguments of the method (via the functional black-box interface) (pg. 29, paragraph 430-434).

As to claim 31, CHONG teaches a memory for storing data for access by a process (application) being executed by a processor, the memory comprising: a structure for defining a value of a parameter associated with an input hashtable (variable to be passed in via the black-box interface / data bindings) (pg. 29, paragraph 430-434; pg. 24, paragraph 353-357), mapping a value of a parameter associated with an output hashtable (variable to be passed out via the blackbox interface / data bindings) (pg. 29, paragraph 430-434; pg. 24, paragraph 353-357), and defining a user interface associated with a custom activity (visual icon representing the component) (pg. 29, paragraph 430-434) that performs a process based on the values of the parameters in the input and output hashtables (via using the adapters to access external data sources) (pg. 23, paragraph 350 – pg. 24, paragraph 352; pg. 24, paragraph 353-354).

As to claims 32 and 33, CHONG teaches a memory for storing data associated with a custom activity for access by a process (application) being executed by a processor the memory comprising: a structure (workflow) specifying an input tag that obtains a value for an input hashtable to be used as an argument in a method (via the setvariableaction tag / modelinterface tag / modelvardef tag / databindinginterface tag / databinding tag wherein a data binding identifies both the input variables and the output variables), specifying an output tag that specify parameters that define what to do with parameters in an output hashtable including output arguments associated with the method (via performing any action tag / databindinginterface tag / databinding tag wherein a data binding identifies both the input variables and the output variables), and specifying design tags that define a user interface associated with the custom activity (listview tag / devicebasedview tag) (pg. 16, paragraph 284 – 297; pg. 24, paragraph 351-357).

As to claim 34, CHONG teaches a memory for storing data for access by a process (application) being executed by a processor, the memory comprising: a structure defining a custom activity (component / object / workflow step) implemented in a process management system (development system / workflow environment) by defining a package (component) (via componentization) (pg. 29, paragraph 431-434) for importing packages external to the process management system (via using the adapters to access external data sources) (pg. 23, paragraph 350 – pg. 24, paragraph

352; pg. 24, paragraph 353-354), defining an init method for defining initialization tasks associated with the custom activity (pre-action tasks on the component) (pg. 15, paragraph 267-271; pg. 12, paragraph 231-234), and defining a perform method for executing tasks associated with the custom activity (action tasks on the component) (pg. 15, paragraph 267-271; pg. 12, paragraph 231-234).

As to claim 37, CHONG teaches a system for creating and implementing custom activities (components / objects / workflow steps) in a process management environment (development system / workflow environment), comprising: a processor; and a memory containing instructions executable by the processor (pg. 8, paragraph 177-178) to: receive a selection to add a custom palette (via componentization packaging the visual icon of a component to be loaded in a palette of another system such that the component is available to the other user) (pg. 29, paragraph 430-434) (pg. 14, paragraph 258); receiving information reflecting an identifier associated with the custom palette (via receiving the Jar file and using the component by dragging and dropping its visual icon); and assigning a visual representation (visual icon) to the custom palette reflecting a custom activity that may be used in an automated workflow process (via componentization packaging the visual icon of a component to be loaded in a palette of another system such that the component is available to the other user) (pg. 29, paragraph 430-434) (pg. 14, paragraph 258) to access information external to the process management environment (via using the adapters to access external data sources) (pg. 23, paragraph 350 – pg. 24, paragraph 352; pg. 24, paragraph 353-354).

As to claim 38, CHONG teaches a system for creation and implementing a custom activity (component / object / workflow step) in a process management environment (development system / workflow environment) comprising: a processor; and a memory containing instructions executable by the processor (pg. 8, paragraph 177-178) to: receive a request to generate a palette associated with the custom activity; assign the custom activity to the palette (via performing componentization wherein the component is added to the IDE by copying icon files into a directory defined by the user such that components are added to the palette) (pg. 29-30, paragraph 433); and determine activation of the custom activity (component) based on a manipulation (drag and drop) associated with the palette pg. 29, paragraph 430-434) (pg. 14, paragraph 258), wherein the custom activity accesses resources external to the process management environment (via using the adapters to access external data sources) (pg. 23, paragraph 350 – pg. 24, paragraph 352; pg. 24, paragraph 353-354).

As to claim 39, CHONG teaches a system for creating an implementing a custom activity (component / object / workflow step) in a process management environment (development system / workflow environment), comprising: a processor; and a memory containing instructions executable by the processor (pg. 8, paragraph 177-178) to: receive a first file (adapter / class) defining with an interface with a package external to the process management system (external data source); receive a second file defining parameters (variables) that the first file uses; archive the first and second file in an

archive file (via receiving the classes and variable information packaged in a Jar file) (via componentization) (pg. 29, paragraph 431-434); and invoke the first and second file based on a manipulation of an image reflecting the custom activity in a visual process map (workflow diagram) reflecting an automated workflow process (via dragging and dropping the icon representing the component to a diagram of the workflow) (pg. 13, paragraph 257 – 258; pg. 14, paragraph 262).

As to claim 2, CHONG teaches the data representation file includes a section that determines the appearance of a representation reflecting the activity (visual icon) (pg. 29, paragraph 431-434).

As to claim 3, CHONG teaches the class file includes a method that is configured (via the adapter bindings and the functional black box interface) to obtain a value of a parameter defined in the data representation file (pg. 29, paragraph 431-434; pg. 24, paragraph 351-357).

As to claim 4, CHONG teaches receiving data that defines a package for the class file; and receiving data that defines methods that retrieve and set values to variables to be used by the activity (via componentizing the behavior of the component and instruction on how to assemble the resources of the component into the application (pg. 29, paragraph 434-434).

As to claim 5, CHONG teaches receiving data that reflects a method that defines variables that are constant across all instances of the activity (via the black-box interface / data bindings that defines both input and output variables) (pg. 29, paragraph 430-434; pg. 24, paragraph 353-357; pg. 30, paragraph 433).

As to claim 6, CHONG teaches the method is associated with an input hashtable to define values of a variable used by the activity (variable to be passed in via the black-box interface / data bindings) (pg. 29, paragraph 430-434; pg. 24, paragraph 353-357; pg. 30, paragraph 433).

As to claim 7, CHONG teaches receiving data (for componentization or at the receiving node of the Jar file) reflecting a method that defines values for variables in a first hashtable and retrieves values for variables from a second hashtable (variable to be passed in and out via the black-box interface / data bindings) (pg. 29, paragraph 430-434; pg. 24, paragraph 353-357; pg. 30, paragraph 433).

As to claim 8, CHONG teaches receiving data reflecting a method that releases resources used by an application that implements the activity when the application is unloaded from the process management system (via the delete action) (pg. 14-15, paragraph 265).

As to claim 9, CHONG teaches receiving data reflecting a first section that defines a type and name of the class file (adapter); receiving data reflecting a second section that defines parameters with values that remain constant within all instances of the activity (via the variables or data bindings provided); receiving data reflecting a third section that sets values for selected parameters within a first hashtable (via an action operation on the variables); receiving data reflecting a fourth section that defines what to do with parameters included in a second hashtable (via instruction on how to assemble the resources in the component into the application); and receiving data reflecting a fifth section associated with a visual representation associated with the activity (visual icon of the component) (pg. 29, paragraph 431-435).

As to claim 10, CHONG teaches packaging the first and second data into one of a JAR file or a ZIP file (pg. 29, paragraph 431).

As to claim 11, CHONG teaches locating the packaged data; and receiving data reflecting a visual representation (visual icon) that corresponds to the packaged files (pg. 29, paragraph 431-434).

As to claim 13, CHONG teaches the file is an archive file and includes the visual representation (pg. 29, paragraph 431-434).

As to claim 15, CHONG teaches the model is an image reflecting the custom activity (visual icon) (pg. 29, paragraph 431-434).

As to claim 16, CHONG teaches packaging the file and model into an archive file (pg. 29, paragraph 431-434).

As to claim 17, CHONG teaches associating the custom activity with a workflow process managed by the process management system (via dragging and dropping the component via its visual icon into the workflow diagram) (pg. 13, paragraph 257 – 258; pg. 14, paragraph 262).

As to claim 18, CHONG teaches determining a position of the model in a visual process map (workflow diagram) reflecting the workflow process; and invoking the custom activity in the workflow process based on the determination (via linking the components of the workflow diagram) (pg. 13, paragraph 257 – 258; pg. 14, paragraph 262).

As to claim 19, CHONG teaches the at least one file includes a Java class file (object oriented class) (pg. 7, paragraph 149; pg. 24, paragraph 351-352; pg. 25, paragraph 369) and an XML description file (pg. 8, paragraph 176; pg. 19, paragraph 325).

As to claim 22, CHONG teaches receiving package information associated with the first file (adapters) that implement packages external to the process management system (via using the adapters to access the external data sources and manipulate the external data sources) (pg. 23, paragraph 350 – pg. 24, paragraph 352; pg. 24, paragraph 353-354).

As to claim 23, CHONG teaches receiving data that interacts with parameters associated with a hashtable defined in the second file (via performing functions to acquire the in variables for the adapter) (pg. 24, paragraph 353-357).

As to claim 24, CHONG teaches receiving data associated with the second file that defines at least one hashtable used by the first file (via defining or retrieving the in variables for the adapter) (pg. 24, paragraph 353-357).

As to claim 25, CHONG teaches the first file reflects a class file (object oriented class) (pg. 7, paragraph 149; pg. 24, paragraph 351-352; pg. 25, paragraph 369) and the second file reflects an XML file (pg. 8, paragraph 176; pg. 19, paragraph 325).

As to claim 26, CHONG teaches archiving the files in a n archive file consisting of one of a JAR file and a ZIP file (pg. 29, paragraph 431-434).

As to claims 28 and 29, CHONG teaches the data representation is an XML description file (pg. 8, paragraph 176; pg. 19, paragraph 325).

As to claim 35, CHONG teaches the perform method is associated with at least a first hashtable (variable) including values corresponding to data fields and a second hashtable including values to be placed in the data fields (pg. 24, paragraph 353-357).

As to claim 36, CHONG teaches the custom activity may have a plurality of instances and wherein the init method defines an association with resources external to the process management system and are shared by all instances of the custom activity (via a component is a container of other components and the components have instructions on how to assemble the resources in the component into the application using that component) (pg. 29, paragraph 431-434).

As to claims 40-65, reference is made to a computer readable medium that corresponds to the method of claims 1-26 and is therefore met by the rejection of claims 1-26 above.

As to claims 66-91, reference is made to a system that corresponds to the method of claims 1-26 and is therefore met by the rejection of claims 1-26 above.

Response to Arguments

3. Applicant's arguments filed October 13, 2006 have been fully considered but they are not persuasive. Applicant argued that Chong does not teach first data and second data because there is no file that contains an encapsulated set of program statements and methods that specify the data and behavior of an object. The examiner disagrees. Chong states that each component created by the system comprises at least: all media imported into the component including strings, images, audio files, and speech recognition grammars; and a description that defines the behavior of the component (pg. 29, paragraph 0431). These components are stored in an archive file format, one of which is a Java JAR file (pg. 29, paragraph 0431). The componentization process opens the JAR file (which is a collection of multiple files that have been concatenated and indexed so that the files can be later extracted into individual files) and writes each of the components into the JAR file (pg. 30, paragraph 0437). Therefore, Chong does teach the class file (a description that defines the behavior of the component) that is received and packaged into a JAR file.

Applicant then argues that Chong merely discloses the contents of each component and does not disclose creating an activity including receiving data such that the receiving data is not inherent in Chong's component. The examiner disagrees. As outlined above, Chong packages all the files of the component's data into a package for distribution to another developer's components directory (see recitations provided above and pg. 30, paragraph 0434). Therefore, Chong teaches the steps of receiving the data and packaging the data.

Applicant argues that Chong does not teach the steps of associating the packaged data with an activity that may be used in an automated workflow process to access information external to the process management system. In particular, the workflow diagram "may include components", but the workflow diagram does not teach the claimed "activity that may be used in an automated workflow process" at least because the workflow diagram cannot be both the claimed activity and the claim "automated workflow process." The examiner disagrees. First the claim limitations indicate that the activity **"may"** be used in an automated workflow process to access external information. This limitation does not require the activity to perform the external access. Secondly, components that are loaded into another developer's system appear on a component palette and may be incorporated into any application built using the present invention simply by dragging the visual icon into the editor window (pg. 30, paragraph 0433). When the components are used in the application, all the items that are part of the component become referenceable by the application (pg. 30, paragraph 0433). An interaction process model is part of the component data and sets forth the user interaction flow between the client device and the server (pg. 29, paragraph 0431). In addition, the application has both a main layer and sub-controller layers of components (pg. 38, paragraph 0529 and 0530). At the very least, the packaged data (component) is associated with one workflow diagram (the subcontroller layer) that is being used in another workflow diagram (main layer). Therefore, Chong teaches the step of associating the packaged data with an activity that may be used to access information external to the process management system.

Applicant then argues that the provisional application of Chong does not teach the associating step. The examiner disagrees. At least at page 30 which defines the process flows and how they are connected to applications, and pages 31 and 41-42, that discloses that flow (process and interaction) is received by the deployment manager and saved to a repository to be deployed provides support for the associating step and the other steps of claim 1. See also page 33 and pages 39-41, which details the functions of submodel controller that creates and executes a submodel that is attached to a master model and that it is possible to logically separate the application into several workflow dimensions or that the workflow diagram is broken down among different layers that are enabled / disabled (main / sub-controller layers of workflow diagrams).

Applicant argues that Chong does not teach or suggest every element of claim 12 because Chong does not teach the step of defining a file associated with a custom activity. The examiner disagrees. As outlined above, Chong teaches the defining and receiving operations by at least associating a component with a subcontroller layer of components of the application that is also associated with a main layer of components of the application. Therefore, the rejection is met as disclosed above.

Applicant argues that Chong does not teach or suggest every element of claim 14 because Chong does not teach or suggest defining a model associated with the custom activity. As outlined above, Chong teaches the defining and receiving operations by at least associating a component with a subcontroller layer of

components of the application that is also associated with a main layer of components of the application. Therefore, the rejection is met as disclosed above.

Applicant argued that Chong does not teach or suggest every element of claim 20 because Chong does not teach or suggest creating an image reflecting a custom activity. As outlined above, Chong teaches associating a component with a subcontroller layer of components of the application that is also associated with a main layer of components of the application. Each component has the interaction process model and a visual icon that represents the component (pg. 29, paragraph 0431). Therefore, Chong teaches creating an image reflecting a custom activity because the icon is deployed and instantiated to represent the deployed component or custom activity and is later invoked based on a manipulation of the image by the user and placed in a process map (via dragging and dropping the image onto the main layer of components to thereby invoke the component) (see page 14, paragraph 0263 and 0264). In addition, the palette allows users to invoke or set actions in the components, including keyboard actions (pg. 15-16, paragraph 0267 – 0283). Therefore, the rejection is met as disclosed above.

Applicant argued that Chong does not teach or suggest every element of claim 21 because Chong does not teach archiving the files in an archive file such that when the custom activity is activated the archived files are accessed and executed. The examiner disagrees. Chong clearly teaches archiving the files in an archive file (pg. 29, paragraph 0431 – 0432; pg. 30, paragraph 0436-0437). Chong also states that components that are loaded into another developer's system appear on a component

palette and may be incorporated into any application built using the present invention simply by dragging the visual icon into the editor window (pg. 30, paragraph 0433). When the components are used in the application, all the items that are part of the component become referenceable by the application (pg. 30, paragraph 0433). An interaction process model is part of the component data and sets forth the user interaction flow between the client device and the server (pg. 29, paragraph 0431). In addition, the application has both a main layer and sub-controller layers of components (pg. 38, paragraph 0529 and 0530). At the very least, the packaged data (component) is either associated with one workflow diagram (the subcontroller layer) or represents a workflow diagram that is being used and invoked by another workflow diagram (main layer). One example of how the claim is met would be that the custom activity is the main layer that is accessing a sub-controller layer to exchange information with external resources (diagram of the sub-controller layer). Another example is the custom activity is the sub-controller layer such that when the component is accessed and executed it's embedded diagram exchanges information with external resources. The claims are broad enough to read upon any number of interpretations.

Applicant argued that Chong does not teach or suggest every element of claim 30 because Chong does not teach or suggest "a first hashtable reflecting data values to be used as input arguments in a method, and a second hashtable reflecting output arguments of the method". The examiner disagrees. Chong teaches a request scope applies to variables that are visible only during the time that the controller is processing a request (pg. 25, paragraph 0361). When the end-user data reaches the controller, the

request scope will begin. The data that is sent back to the controller it is in the form of name-value pairs. Therefore, at least here, Chong teaches a first hashtable reflecting data values to be used as input arguments in a method (request scope) and a second hastable reflection output arguments of the method (name value pairs). In addition, see page 33, Submodel Controller wherein there exists an inputMap property object and outputMap property. This is at least supported on page 72, of the provisional application wherein the engine records the name-value pairs submitted with the request and records the response returned.

Applicant argued that Chong does not teach or suggest every element of claim 31 because Chong does not teach or suggest an input hashtable, mapping a value of a parameter associated with an output hashtable, and defining a user interaction associated with a custom activity that performs a process based on the values of the parameters in the input and output hastables. The examiner disagrees. As outlined above regarding claim 30, Chong teaches an input hastable and mapping a value of a parameter associated with an output hasthable and which is supported in the provisional application. Chong further teaches a user interface associated with a custom activity that can be invoked as outlined above in relation to the icon being representing the component and being drag and dropped into an application and thereby invoked. Therefore, Chong would teach all of the limitations of the claims as outlined since the image is dragged and dropped thereby invoking an action by a controller using the input and thereby mapping the received output to the input.

Applicant argued that Chong does not teach or suggest every element of claim 32 because Chong does not teach or suggest the "input hashtable" and the "output hashtable" and "specifying design tags that define a user interface associated with the custom activity." The examiner disagrees. As outlined above regarding claim 30, Chong teaches an input hastable and mapping a value of a parameter associated with an output hasthable and which is supported in the provisional application. Chong further teaches that each component / sub-controller comprises the presentation layer defined by the presentation design module (pg. 29, paragraph 0431). The presentation design module specifies tags related to the view, e.g. definition of a user interface to the component (multiple paragraphs throughout pages 18-21). Therefore, Chong teaches the claim limitations as disclosed. This is at least supported in the provisional application on page 35-36, Interaction Manager and Class Structure.

Applicant argued that Chong does not teach or suggest every element of claim 34 because Chong does not teach or suggest the claimed "custom activity" or "defining an init () method for defining initialization tasks associated with the custom activity, and defining a perform method for executing task associated with the custom activity. The examiner disagrees. The examiner has equated Chong's "pre-actions" as the initialization task associated with the custom activity because these actions are performed before any other actions are performed, and "actions" as the perform methods for executing tasks associated with the custom activity because these operations are performed the component is invoked and all pre-actions are performed. Chong teaches that the creation of the interaction process model starts off with an

empty controller called start model wherein each controller has a start state and at least one end state (pg. 36, paragraph 0522-0523). Like all states the start state contains pre-actions, actions and post-actions which are all represented by the <Property> element (pg. 36, paragraph 0523). Transitions are next created to indicate when to perform the actions, for instance when transition from a start state to a next state, the cited pre-action, action, and post-action are performed (pg. 37-38, paragraphs 0526-0529). Therefore, the interaction process model or component that has the model (custom activity) defines pre-actions, actions and post actions when transitioning between the different states and referencing external information in servers. This is at least supported in the provisional application on page 32 (default controller) and pg. 33, Submodel Controller.

Applicant argued that Chong does not teach or suggest every element of claims 37 and 38 because Chong does not teach or suggest the claimed "a custom palette" and "an identifier associated with the customer palette". The examiner disagrees. Chong teaches the component palette consists of two parts, a component category selector and a component list (pg. 31, paragraph 0438). Developers can specify the category in which a component should be created to better organize sets of related components. By specifying the category in which the component should be created and selecting a specific category to view those components for adding to an application, developers are inherently selecting to add a custom palette and provide information about a category of the components for visual manipulation (thereby the palette receiving information reflecting an identifier of a category of components to be viewed

and possibly added to the application). Therefore, Chong teaches the limitations of the claims and the rejection is maintained above. This is supported in the provisional application at least at pg. 41-42, Mobile Application Componentization wherein Once created, components are saved in a component repository and exposed in the Studio represented by the visual icon such that components may be incorporated into another application by simply dragging the visual icon into the process design window of the application and page 29, wherein the process designer uses the Covigo Studio Process Designer to identify each of the states needed for the applications logical process flow and has the option of defining new states or reusing already defined components by dragging and dropping them from the menu. For at least the following reasons the rejections to the claims are maintained.

Conclusion

4. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of

the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lewis A. Bullock, Jr. whose telephone number is (571) 272-3759. The examiner can normally be reached on Monday-Friday, 8:30 a.m. - 5:00 p.m..

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

January 22, 2007

LEWIS A. BULLOCK, JR.
PRIMARY EXAMINER